자료구조 보고서

Homework#4

학과 : 소프트웨어학과
학번 : 2016039040
이름 : 윤용진

matrix.c 코드를 수정/편집하여 과제를 진행.

23. 프로그램 보고서에 본인의 GitHub URL을 명시한다.
https://github.com/uniz21/DataSturcture_Homework-4

1. 행렬의 행(row)과 열(column)의 값을 입력 받는다. (scanf() 사용)
2. 입력받은 행(row)과 열(column)의 값을 함수구현에 활용한다.
3. 모든 행렬은 동적 메모리할당(dynamic memory allocation) 방식으로 생성한다.
(함수: create matrix())
5. Initialize Matrix - 행렬 A, B의 성분값(entry value)은 0 ~ 19 사이의 값을 갖는 랜덤값(random
value)으로 채운다. (rand() 사용) (함수: fill data())
6. Print Matrix - A와 B 행렬을 출력한다. (print matrix() 사용)
7. Add Matrix - A + B를 구현한다. (함수: addition matrix())
8. Subtract Matrix - A − B를 구현한다. (함수: subtraction matrix())
9. Transpose matrix a - A의 전치행렬 T를 구현한다. (함수: transpose matrix())
10. Multiply Matrix - A × T를 구현한다. (함수: multiply matrix())
11. Quit를 제외한 모든 메뉴는 다시 실행될 수 있어야 한다.
소스코드

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


/* Method Declaration */
int** create_matrix(int row, int col);
void print_matrix(int** matrix, int row, int col);
int free_matrix(int** matrix, int row, int col);
int fill_data(int** matrix, int row, int col);
int addition_matrix(int** matrix_a, int** matrix_b, int row, int col);
int subtraction_matrix(int** matrix_a, int** matrix_b, int row, int col);
int transpose_matrix(int** matrix, int** matrix_t, int row, int col);
int multiply_matrix(int** matrix_a, int** matrix_t, int row, int col);

int main()
{

    char command;
    printf("[----- [윤용진]  [2016039040] -----]\n");
```

```
    int row, col;
    srand(time(NULL));

    printf("Input row and col : ");
    //행과 열의 크기를 입력받아 행렬을 생성
    scanf("%d %d", &row, &col);
    int** matrix_a = create_matrix(row, col);
    int** matrix_b = create_matrix(row, col);
    int** matrix_a_t = create_matrix(col, row);

    printf("Matrix Created.\n");

    //행렬이 비어있을 경우 비정상 종료
    if (matrix_a == NULL || matrix_b == NULL) { return -1; }

    do {

printf("----------------------------------------------------------------------\n");
        printf("                    Matrix Manipulation                         \n");

printf("----------------------------------------------------------------------\n");
        printf(" Initialize Matrix   = z          Print Matrix        = p \n");
        printf(" Add Matrix          = a          Subtract Matrix     = s \n");
        printf(" Transpose matrix_a  = t          Multiply Matrix     = m \n");
        printf(" Quit                = q \n");

printf("----------------------------------------------------------------------\n");

        printf("Command = ");
        scanf(" %c", &command);

        switch (command) {
        case 'z': case 'Z'://행렬 초기화
            //각 행렬의 크기는 유지한 채로 새로운 행렬을 생성한다.
            printf("Matrix Initialized\n");
            fill_data(matrix_a, row, col);
            fill_data(matrix_b, row, col);
            break;
        case 'p': case 'P'://행렬 출력
            printf("Print matrix\n");
            printf("matrix_a\n");
            print_matrix(matrix_a, row, col);
            printf("matrix_b\n");
```

```
            print_matrix(matrix_b, row, col);
            break;
        case 'a': case 'A'://행렬의 합
            printf("Add two matrices\n");
            addition_matrix(matrix_a, matrix_b, row, col);
            break;
        case 's': case 'S'://행렬의 차
            printf("Subtract two matrices \n");
            subtraction_matrix(matrix_a, matrix_b, row, col);
            break;
        case 't': case 'T'://전치행렬
            printf("Transpose matrix_a \n");
            printf("matrix_a\n");
            transpose_matrix(matrix_a, matrix_a_t, col, row);
            break;
        case 'm': case 'M'://행렬A와 A의 전치행렬T의 곱
            //전치행렬T 생성
            printf("matrix_t\n");
            transpose_matrix(matrix_a, matrix_a_t, col, row);
            printf("Multiply matrix_a with transposed matrix_a \n");
            multiply_matrix(matrix_a, matrix_a_t, row, col);
            break;
        case 'q': case 'Q'://종료
            printf("Free all matrices..\n");
            free_matrix(matrix_a_t, col, row);
            free_matrix(matrix_a, row, col);
            free_matrix(matrix_b, row, col);
            break;
        default:
            printf("\n       >>>>>   Concentration!!   <<<<<      \n");
            break;
        }

    } while (command != 'q' && command != 'Q');

    return 1;
}

/* create a 2d array whose size is row x col using malloc() */
int** create_matrix(int row, int col)
{
    /* check pre conditions */
    // 행과 열의 크기 확인
```

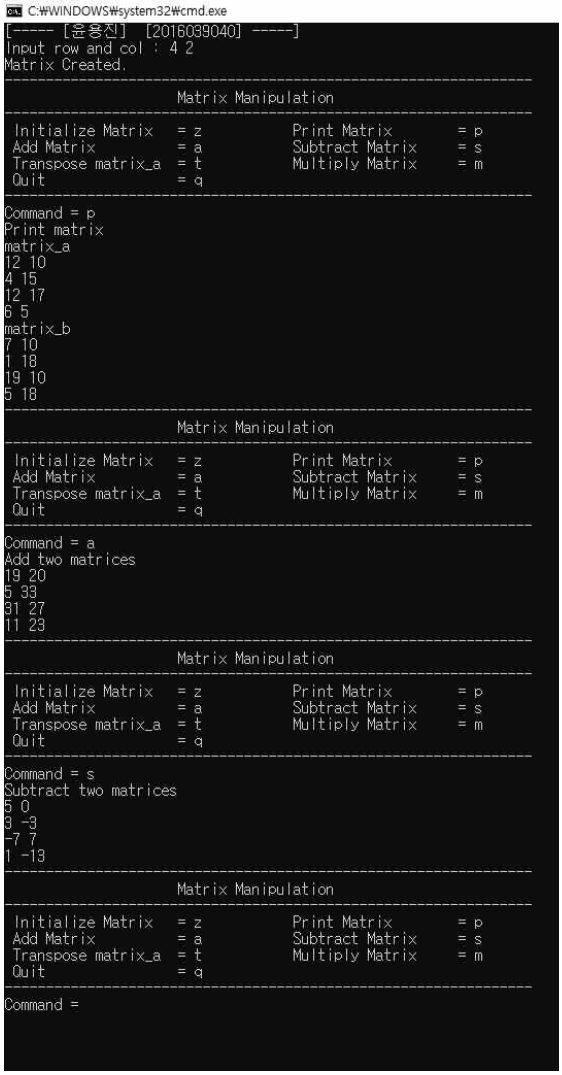
```
    if (row <= 0 || col <= 0) {
        printf("Check the sizes of row and col!\n");
        return NULL;
    }

    //동적 메모리 할당으로 행렬 생성
    int **x;
    x=malloc(row * sizeof(*x));
    for (int i = 0; i < row; i++)
        x[i]=malloc(col * sizeof(**x));
    //행렬채우기
    fill_data(x, row, col);
    //행렬 반환
    return x;
}

/* print matrix whose size is row x col */
void print_matrix(int** matrix, int row, int col)
{
    /* check pre conditions */
    if (matrix == NULL) {
        printf("Matrix broken");
        return -1;
    }

    //행*열 만큼 반복하여 성분값 출력
    for (int i = 0; i < row; i++) {
        for (int k = 0; k < col; k++) {
            printf("%d ", matrix[i][k]);
        }
        printf("\n");
    }
}


/* free memory allocated by create_matrix() */
int free_matrix(int** matrix, int row, int col)
{
    //동적 메모리할당 해제
    for (int i = 0; i < row; i++)
            free(matrix[i]);
    free(matrix);
    return 1;
```

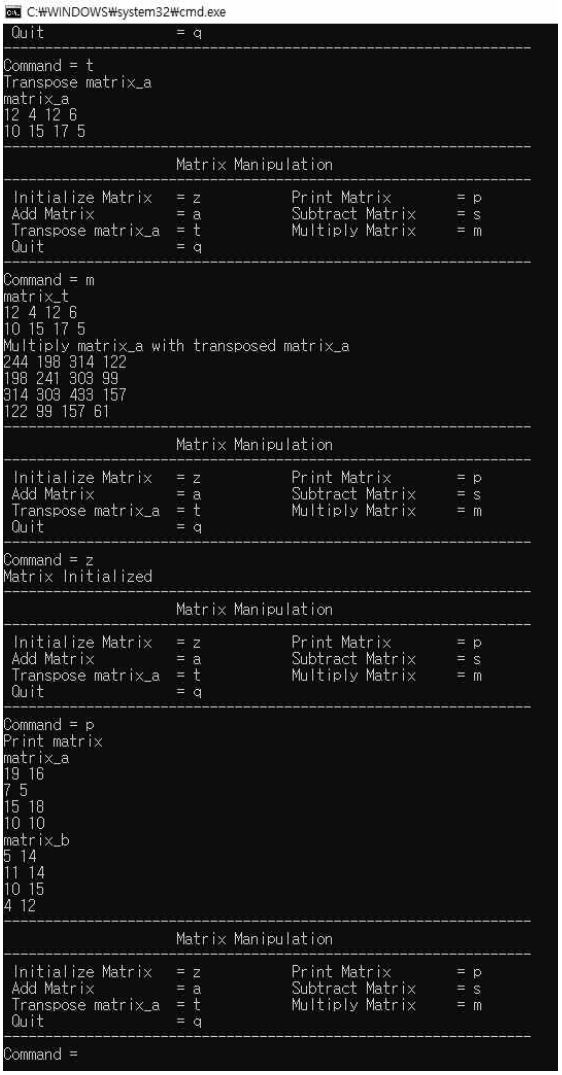
```
}


/* assign random values to the given matrix */
int fill_data(int** matrix, int row, int col)
{
    //행*열의 크기만큼 반복
    for (int i = 0; i < row; i++)
    {
        for (int k = 0; k < col; k++)
            //0~19 사이의 난수값 저장
            matrix[i][k] = rand()%20;
    }
    return 1;
    /* check post conditions */
    if (matrix == NULL) {
        printf("Matrix broken");
        return -1;
    }
}

/* matrix_sum = matrix_a + matrix_b */
int addition_matrix(int** matrix_a, int** matrix_b, int row, int col)
{
    int** matrix_sum = create_matrix(row, col);
    //행*열의 크기만큼 반복
    for (int i = 0; i < row; i++)
    {
        for (int k = 0; k < col; k++)
        {
            //A행렬과 B행렬의 각각 대응하는 성분의 합
            matrix_sum[i][k] = matrix_a[i][k] + matrix_b[i][k];
        }
    }
    print_matrix(matrix_sum, row, col);
    /* check post conditions */
    if (matrix_sum == NULL) {
        printf("Add Failed");
        return -1;
    }
    return 1;
}
```

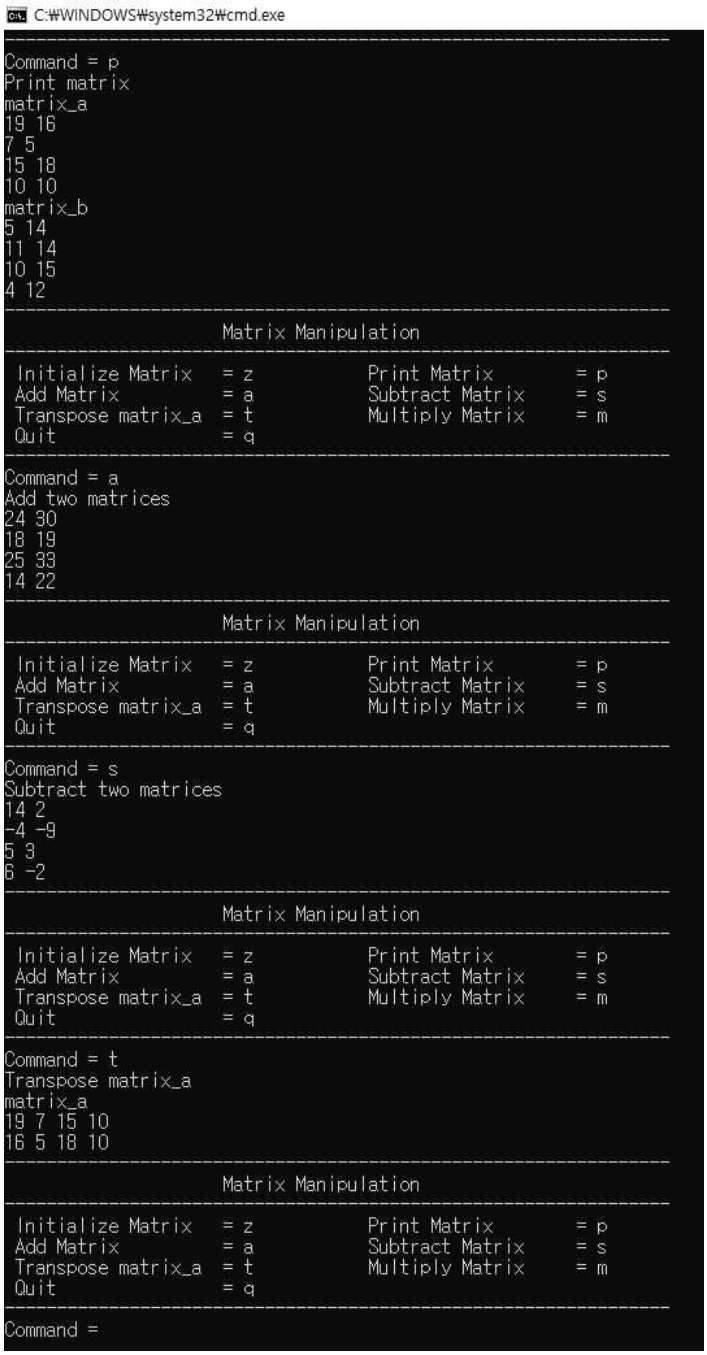
```c
/* matrix_sub = matrix_a - matrix_b */
int subtraction_matrix(int** matrix_a, int** matrix_b, int row, int col)
{
    int** matrix_sub = create_matrix(row, col);
    //행*열의 크기만큼 반복
    for (int i = 0; i < row; i++)
    {
        for (int k = 0; k < col; k++)
        {
            //A행렬과 B행렬의 각각 대응하는 성분의 차
            matrix_sub[i][k] = matrix_a[i][k] - matrix_b[i][k];
        }
    }
    print_matrix(matrix_sub, row, col);
    /* check post conditions */
    if (matrix_sub == NULL) {
        printf("Sub Failed.");
        return -1;
    }
    return 1;
}

/* transpose the matrix to matrix_t */
int transpose_matrix(int** matrix, int** matrix_t, int row, int col)
{
    //행*열의 크기만큼 반복
    for (int i = 0; i < row; i++)
    {
        for (int k = 0; k < col; k++)
        {
            //전치행렬은 행과 열이 반대이다.
            matrix_t[i][k] = matrix[k][i];
        }
    }
    print_matrix(matrix_t,row,col);
    /* check post conditions */
    if (matrix_t == NULL) {
        printf("Transpose Failed.");
        return -1;
    }
    return 1;
}
```

```c
/* matrix_axt - matrix_a x matrix_t */
int multiply_matrix(int** matrix_a, int** matrix_t, int row, int col)
{
    /* check pre conditions */
    //전치행렬 T가 생성되지 않은 경우
    if (matrix_t == NULL) {
        printf("Check about Transpose Matrix A");
        return -1;
    }
    //행렬 A와 A의 전치행렬 T의 곱이므로 전처리로 행렬의 크기를 비교할 필요는 없다.

    int** matrix_axt = create_matrix(row, row);
    int sum=0;
    //행렬의 곱 규칙에 따라 앞 행렬의 행*행의 크기를 가진 행렬이 생성된다.
    for (int i = 0; i < row; i++)
    {
        for (int k = 0; k < row; k++)
        {
            //앞 행렬의 행을 뒷 행렬의 행에 곱할 때의 연산
            for (int t = 0; t < col; t++)
            {
                sum+=matrix_a[i][t] * matrix_t[t][k];
            }
            matrix_axt[i][k] = sum;
            sum = 0;
        }
    }
    print_matrix(matrix_axt,row,row);

    /* check post conditions */
    if (matrix_axt == NULL) {
        printf("Multiply Failed.");
        return -1;
    }
    return 1;
}
```

실행결과


C:₩WINDOWS₩system32₩cmd.exe
[----- [윤용진] [2016039040] -----]
Input row and col : 4 2
Matrix Created.
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command = p
Print matrix
matrix_a
12 10
4 15
12 17
6 5
matrix_b
7 10
1 18
19 10
5 18
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command = a
Add two matrices
19 20
5 33
31 27
11 23
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command = s
Subtract two matrices
5 0
3 -3
-7 7
1 -13
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command =


C:₩WINDOWS₩system32₩cmd.exe
Quit = q
Command = t
Transpose matrix_a
matrix_a
12 4 12 6
10 15 17 5
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command = m
matrix_t
12 4 12 6
10 15 17 5
Multiply matrix_a with transposed matrix_a
244 198 314 122
198 241 303 99
314 303 433 157
122 99 157 61
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command = z
Matrix Initialized
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command = p
Print matrix
matrix_a
19 16
7 5
15 18
10 10
matrix_b
5 14
11 14
10 15
4 12
Matrix Manipulation
Initialize Matrix = z Print Matrix = p
Add Matrix = a Subtract Matrix = s
Transpose matrix_a = t Multiply Matrix = m
Quit = q
Command =


C:₩WINDOWS₩system32₩cmd.exe
Command = p
Print matrix
matrix_a
19 16
7 5
15 18
10 10
matrix_b
5 14
11 14
10 15
4 12
Matrix Manipulation
Initialize Matrix = z
Print Matrix = p
Add Matrix = a
Subtract Matrix = s
Transpose matrix_a = t
Multiply Matrix = m
Quit = q
Command = a
Add two matrices
24 30
18 19
25 33
14 22
Matrix Manipulation
Initialize Matrix = z
Print Matrix = p
Add Matrix = a
Subtract Matrix = s
Transpose matrix_a = t
Multiply Matrix = m
Quit = q
Command = s
Subtract two matrices
14 2
-4 -9
5 3
6 -2
Matrix Manipulation
Initialize Matrix = z
Print Matrix = p
Add Matrix = a
Subtract Matrix = s
Transpose matrix_a = t
Multiply Matrix = m
Quit = q
Command = t
Transpose matrix_a
matrix_a
19 7 15 10
16 5 18 10
Matrix Manipulation
Initialize Matrix = z
Print Matrix = p
Add Matrix = a
Subtract Matrix = s
Transpose matrix_a = t
Multiply Matrix = m
Quit = q
Command =